HEADS

PhD PROGRAMME IN HEALTH DATA SCIENCE

Data Synopses

Keeping (just) the necessary data

Computational Statistics

PhD Programme in Health Data Science

Pedro Pereira Rodrigues

U.PORTO
FMUP FACULDADE DE MEDICINA UNIVERSIDADE DO PORTO
MEDCIDS
DEPARTAMENTO DE MEDICINA DA COMUNIDADE, INFORMAÇÃO E DECISÃO EM SAÚDE
CINTESIS
Health. Research.

Computational Statistics

“Computational statistics is what statisticians do with the computer.”

Naeve P. (2000)

Computational Statistics

"Computational statistics is a branch of mathematical sciences concerned with efficient methods for obtaining numerical solutions to statistically formulated problems."

Nickel C. (2020)

Wealth of Health Data

"The routine operation of modern healthcare systems produces a wealth of data in electronic health records, administrative databases, clinical registries, and other clinical systems."

Peek & Rodrigues (2018)

Raw Data

"The convergence of computing and communication has produced a society that feeds on information. Yet most of the information is in its raw form: data."

Witten & Frank (2000)

Knowledge Discovery

"It is widely acknowledged that there is great potential for utilizing these routine data for health research to derive new knowledge about health, disease, and treatments."

Peek & Rodrigues (2018)

Problems with data

"There are a lot of small data problems that occur in big data. They do not disappear because you have got lots of the stuff. They get worse."

Spiegelhalter D (2014)


BIG DATA
with 8 V's
01
VOLUME
Can you find the information you are looking for?
02
VALUE
Can you find it when you most need it?
03
VERACITY
Are you dealing with information or disinformation?
04
VISUALISATION
Can you make sense at a glance? Does it trigger a decision?
05
VARIETY
Is a picture worth a thousand words in 70 languages? Is your information balanced?
06
VELOCITY
Information gains momentum and crises & opportunities evolve in real time. How is outlook for today?
07
VISCOSITY
Does it stick with you? Does it call for action?
08
VIRALITY
AHA to-go? Does it convey a message that can be pasted into a presentation or Instgrammed?


01
VOLUME
Can you find the information you are looking for?
02
VALUE
Can you find it when you most need it?
03
VERACITY
Are you dealing with information or disinformation?
04
VISUALISATION
Can you make sense at a glance? Does it trigger a decision?
05
VARIETY
Is a picture worth a thousand words in 70 languages? Is your information balanced?
06
VELOCITY
Information gains momentum and cri-ses & opportunities evolve in real time. How is outlook for today?
07
VISCOSITY
Does it stick with you? Does it call for action?
08
VIRALITY
AHA to-go? Does it convey a message that ca be pasted into a presentation or Instgrammed?
BIG DATA
with 8 V's

Data Synopses

“Snippets of data that allow more complex, yet efficient, computations.”

Rodrigues P. P. (2020)

Data Synopses

We are going to cover:

- Sufficient statistics
- Histograms
- Micro-clusters
- Fading statistics

Data Streams

“Many sources produce data continuously. These continuous flows of data are called data streams.”

Muthukrishnan S. (2005)

Data Streams vs Data Sets

Data	Set	Stream
Size	finite	infinite
Distribution	i.i.d.	non i.i.d.
Ordering	independent	dependent
Evolution	static	non-stationary

Gama J. & Rodrigues P.P. (2007)

Processing and Learning from Data Streams

"In these settings, traditional machine learning algorithms are obsolete."

Gama J., Sebastião R. & Rodrigues P.P. (2013)

Learning from Data Streams

Algorithms that learn from data streams:

- must process examples at the rate they arrive
- use a single scan and fixed memory
- maintain a decision model at any time
- adapt to the most recent data
- might end-up with approximate results

Data Stream Models

Different data stream models exist:

- insert-only or time series model
  - once an observation x is produced, it cannot be changed
- insert-delete or turnstile model
  - observations x can be deleted or updated
- accumulative or cash-register model
  - each observation x is an increment to a sum X(t) = X(t-1) + x

Data Stream Management

"The main issue in data streams are blocking operators, i.e. queries that require the entire input to be available before they can give an exact output."

Gama J. & Rodrigues P.P. (2007)

Illustrative example

Find the maximum value in a sliding window over a sequence of numbers.

- When we can **store in memory** all the elements of the sliding window, the **problem is trivial** and we can find the exact solution.
- When the size of the window is greater than the available memory, there is **no algorithm** that provides an exact solution!

Open Research Questions

- Approximate query processing techniques to evaluate queries that require an unbounded amount of memory.

- Sliding window query processing, both as an approximation technique and as an option in the query language.

- Sampling to handle situations where the flow rate of the input stream is faster than the query processor.

- The meaning and implementation of blocking operators (e.g., aggregation and sorting) in the presence of unending streams.

Simple Counting Problems

- Given a stream of bits (0's and 1's), maintain a count of the number of 1's in the last N elements seen from the stream.
- Given a stream of elements that are positive integers in the range [0, . . . , R], maintain at every instance the sum of the last N elements seen from the stream.
- Find the number of distinct values in a stream of values with domain [1,...,N].

All these problems have an **exact solution if we have enough memory** to store all the elements in the sliding window. That is, the exact solution requires O(N) space.

Suppose we have restricted memory.

How can we solve these problems using less space than O(N)?

Approximate Computations

Approximate answers are useful if the associated error is in an admissible boundary.

We can define the approximation methods in the data streams as:

(ε,δ)-approximation schema

Given any positive number **ε < 1** and **δ < 1** compute an estimate that,

with probability 1 − δ, is within relative error ≤ ε.

Time and space required to compute an answer depends on ε and δ.

Trade-off: summary size vs approximation error.

Statistical Inequalities

An estimator is a function of the observable sample data that is used to estimate an unknown population parameter. We are particularly interested in interval estimators that compute an interval for the true value of the parameter associated with a confidence $1 - \delta$.

Two types of intervals are:

- **Absolute approximation**: $X - \varepsilon \leq \mu \leq X + \varepsilon$, where ε is the absolute error; and
- **Relative approximation**: $(1 - \delta)X \leq \mu \leq (1 + \delta)X$, where δ is the relative error.

supp reading: Gama J. & Rodrigues P. (2007) “Data Stream Processing”, in *Learning from Data Streams*, Springer Verlag.

Statistical Inequalities

Chernoff and Hoeffding bounds from statistical theory are useful in most of the cases:

- They are **independent from the distribution** generating examples.
- They are **applicable in all situations where observations are independent and generated by a stationary distribution.**
- They are **conservative**, that is they require more observations than when using distribution dependent bounds.

The Chernoff bound is **multiplicative** and its error is expressed as a **relative approximation**.

The Hoeffding bound is **additive** and the error is **absolute**.

While the Chernoff bound uses the sum of events and requires the expected value for the sum, the Hoeffding bound uses the expected value and the number of observations.

supp reading: Gama J. & Rodrigues P. (2007) "Data Stream Processing", in *Learning from Data Streams*, Springer Verlag.

Sufficient Statistics

For most computations, basic statistics suffice:

- counts
- sums
- sums of squares

Some examples follow…

Central tendency

Recursive computation of the **sample mean**:

$$\bar{x}_i = \frac{(i-1) \times \bar{x}_{i-1} + x_i}{i}$$

In fact, to incrementally compute the mean of a variable, we need only to maintain in memory the **number of observations** and the **sum of the values seen so far.**

Dispersion

Simple mathematics allow us to define an incremental version of the **standard deviation**:

$$\sigma_i = \sqrt{\left(\sum x_i^2 - \left(\sum x_i\right)^2 / i\right) / (i-1)}.$$

In that case we need to store three quantities: the **number of data points**, the **sum of data points**, and the **sum of squares of the data points**.

Correlation

Another useful measure that can be recursively computed is the **correlation coefficient**:

$$\mathrm{corr}(a,b) = \frac{\sum(x_i \times y_i) - \frac{\sum x_i \times \sum y_i}{n}}{\sqrt{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}\sqrt{\sum y_i^2 - \frac{(\sum y_i)^2}{n}}}$$

Given two streams x and y, we need to maintain the **sum of each stream**, the **sum of the squared values**, and the **sum of the cross-product**.

Incrementing counters

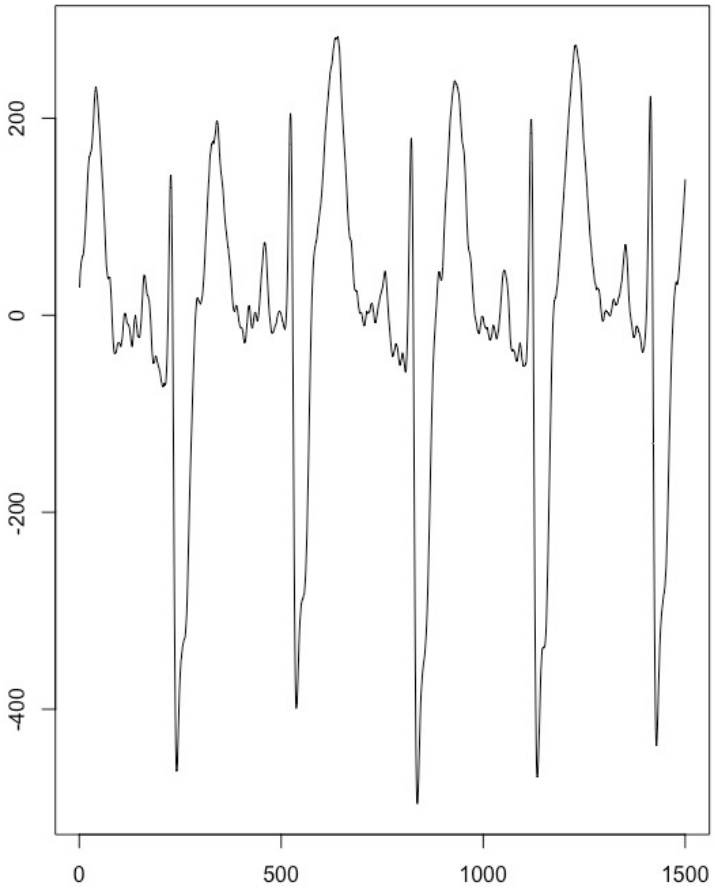
Incrementing counters can be seen as a **sum over a stream of 0's and 1's**, where:

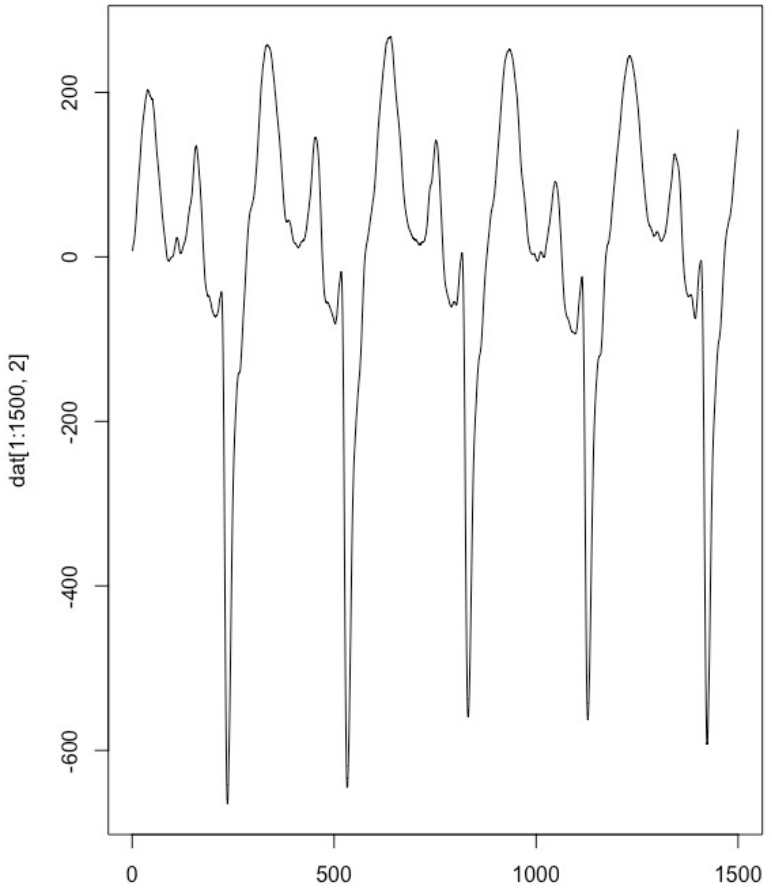
- each 1 represents a hit in the counter, and
- each 0 represents a miss.

This analogy will be extremely important later.

Counters are useful for other statistics computations, but also have their own value, e.g. maintaining **histograms** and **micro-clusters**.

2-dimensional stream





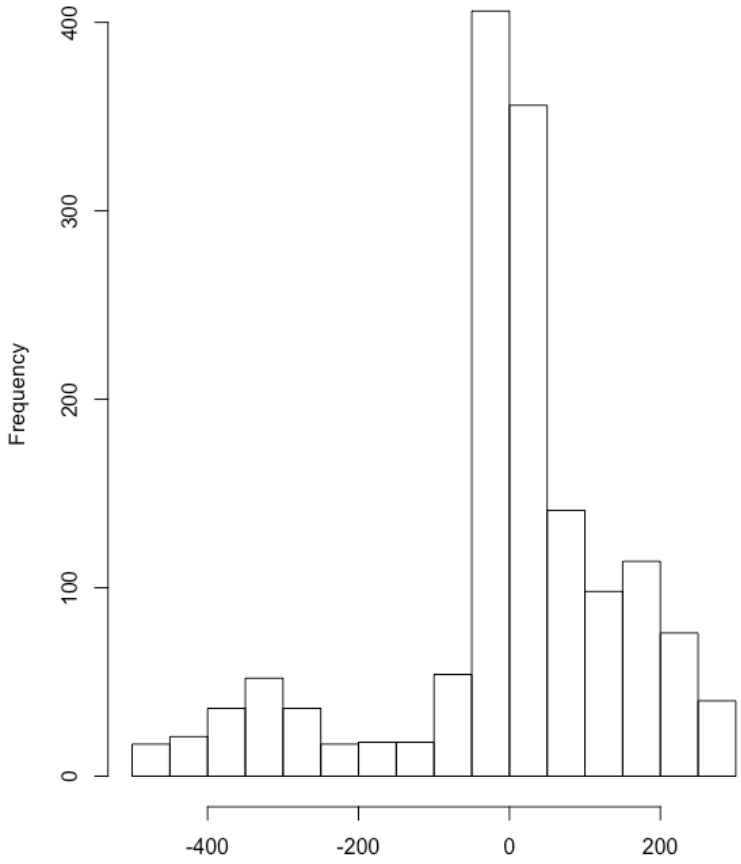
Histograms

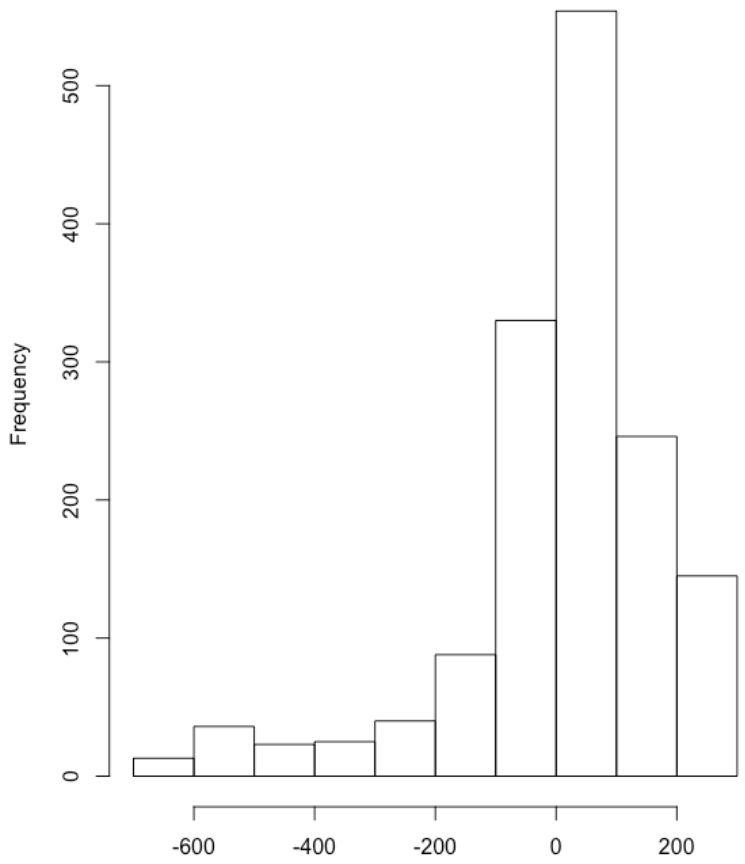
Histograms are distribution representations of individual continuous variables.

- The domain of the variable is divided into contiguous intervals.
- Each interval keeps a counter of the number of data points that fall in it.

Intervals might be defined by width (**equal-width** histograms) or number of points (**equal-frequency** histograms).

Histograms


Frequency
0
100
200
300
400
-400
-200
0
200


Frequency
0
100
200
300
400
500
-600
-400
-200
0
200

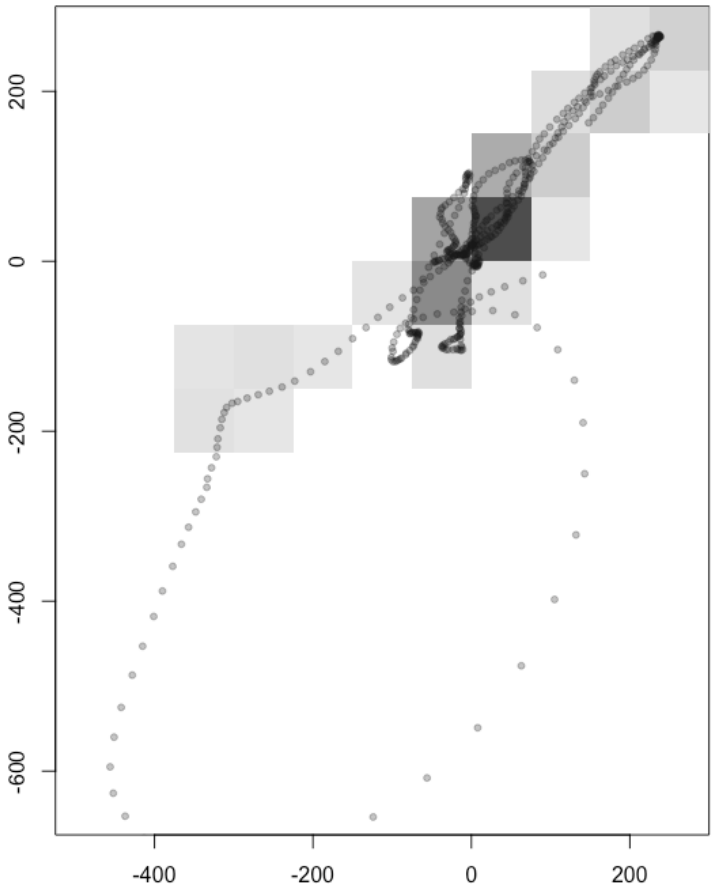
Micro-Clusters

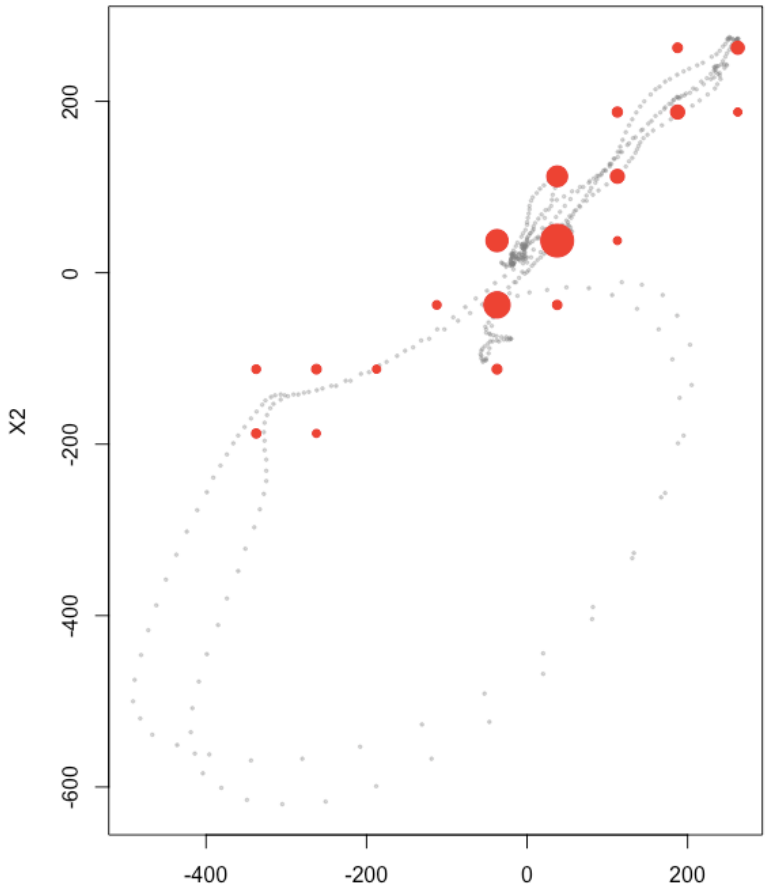
Micro-clusters are simplified representations of dense regions of the data space.

- Most commonly defined by n-dimensional grid cells.
- Each cell keeps a counter of the number of data points that fall in it
- A micro-cluster is a grid cell whose counter exceeds a certain threshold.

Cells might be defined by width (**equal-width** cells) or number of points (**equal-frequency** cells).

Micro-Clusters





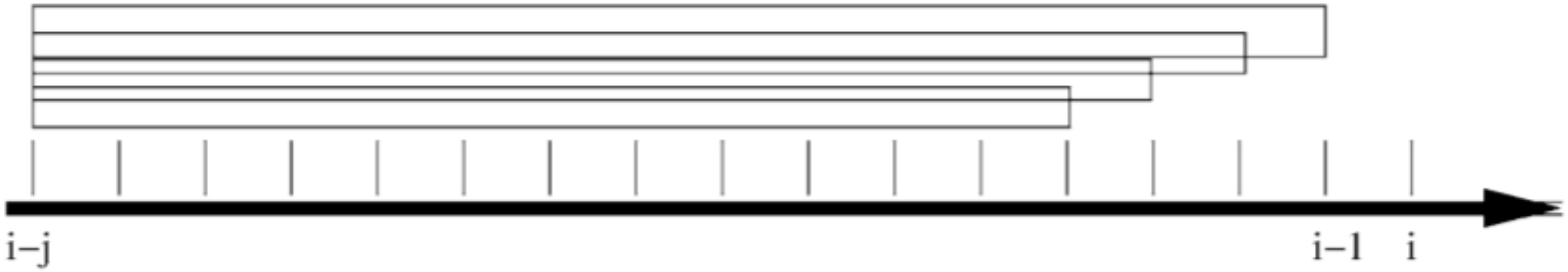
Other statistics

Quantiles, median and other statistics used in range queries can be computed from histograms, and other summaries.

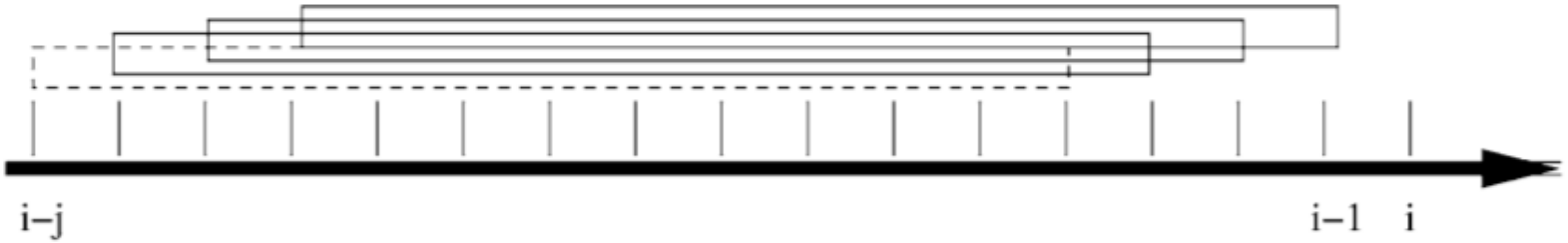
But, in most applications **recent data is the most relevant one**.

To fulfil this goal, a popular approach consists of **defining a time window** covering the most recent data.

Windows



(a) Landmark Window



(b) Sliding Window

Problems with time windows

- Suppose we want to maintain the standard deviation of the values of a data stream using only the last 100 examples, that is in a fixed time window of dimension 100.
- After seeing observation 1000, the observations inside the time window are $x_{901},\ldots,x_{1000}$ and the sufficient statistics are

$$A = \sum_{i=901}^{1,000} x_i;\ \mathrm{B}=\sum_{i=901}^{1,000} x_i^2$$

- Whenever the 1,001th value is observed, the time window moves 1 observation, forgets observation 901 and the updated sufficient statistics are:

$$A = A + x_{1,001} - x_{901} \qquad B = B + x_{1,001}^2 - x_{901}^2$$

- Due to the necessity to forget old observations, **we need to maintain in memory all the observations inside the window.** The same problem applies for time windows whose size changes with time.

Sampling

- Sampling involves loss of information: some tuples are selected for processing while others are skipped.
- If the rate of arrival data in the stream is higher than the capacity to process data, sampling is used as a method to slow down data.
- Another advantage is the use of offline procedures to analyse data, eventually for different goals.
- Nevertheless, sampling must be done in a principled way in order to avoid missing relevant information.

Reservoir Sampling

The classic algorithm to **maintain an online random sample:**

- The basic idea consists of maintaining a sample of size s, called the reservoir.
- As the stream flows, every new element has a certain probability of replacing an old element in the reservoir.

Sampling is a general method for solving problems with huge amounts of data, and has been used in most streaming problems. The literature contains a wealth of algorithms for computing quantiles and distinct counts using samples. However, negative results exist for uniform sampling, mostly for queries involving joins.

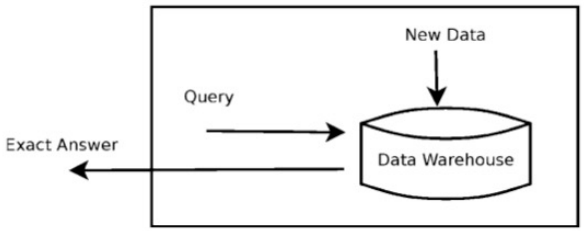
As a matter of fact, sampling works by providing a **compact description of much larger data.**

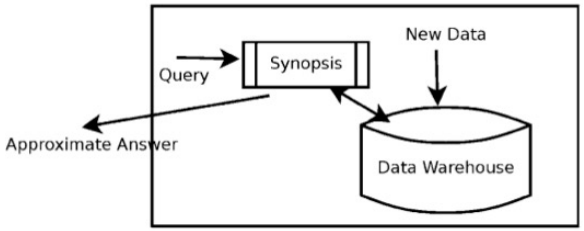
Alternative ways to obtain compact descriptions include **data synopsis.**

Data Synopsis

Multiple **data synopses** exist in literature:

- Frequency moments
- Hash sketches
- Exponential histograms
- Wavelets





supp reading: Gama J. & Rodrigues P. (2007) “Data Stream Processing”, in *Learning from Data Streams*, Springer Verlag.

Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream.

Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream.

```r
# keep sample mean

# definitions
itera <- 2000
set.seed(523)
stream1 <- DSD_Gaussians(k=1, d=1, mu=0, sigma=2)
stream2 <- DSD_Gaussians(k=1, d=1, mu=10, sigma=2)

# initialization
n <- sum <- 0
plot(NULL, xlab="Iteration",ylab="Value", xlim=c(0,itera),
     ylim=c(-3,13), main="Sample Mean")

# iterative step
for (i in 1:itera){
  xi <- ifelse(i<=1000,
               get_points(stream1)[1,1],
               get_points(stream2)[1,1])
  sum <- sum + xi
  n <- n + 1
  points(x=i, y=xi, pch=20, cex=.1, col="black")
  points(x=i, y=sum / n, pch=20, cex=.5, col="red")
}
```

Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream **over a sliding window**.

Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream **over a sliding window**.

```
# keep sample mean over a sliding window

# definitions
itera <- 2000
set.seed(523)
stream1 <- DSD_Gaussians(k=1, d=1, mu=0, sigma=2)
stream2 <- DSD_Gaussians(k=1, d=1, mu=10, sigma=2)
w <- 100

# initialization
n <- sum <- 0
plot(NULL, xlab="Iteration",ylab="Value", xlim=c(0,itera),
     ylim=c(-3,13), main="Sliding Mean")
window <- NULL

# iterative step
for (i in 1:itera){
  xi <- ifelse(i<=1000,
               get_points(stream1)[1,1],
               get_points(stream2)[1,1])
  sum <- sum + xi
  n <- n + 1

  window <- c(window, xi)

  if (length(window) >  w) {
    sum <- sum - window[1]
    n <- n - 1
    window <- window[-1]
  }

  points(x=i, y=xi, pch=20, cex=.1, col="black")
  points(x=i, y=sum / n, pch=20, cex=.5, col="red")
}
```

Catastrophic Forgetting

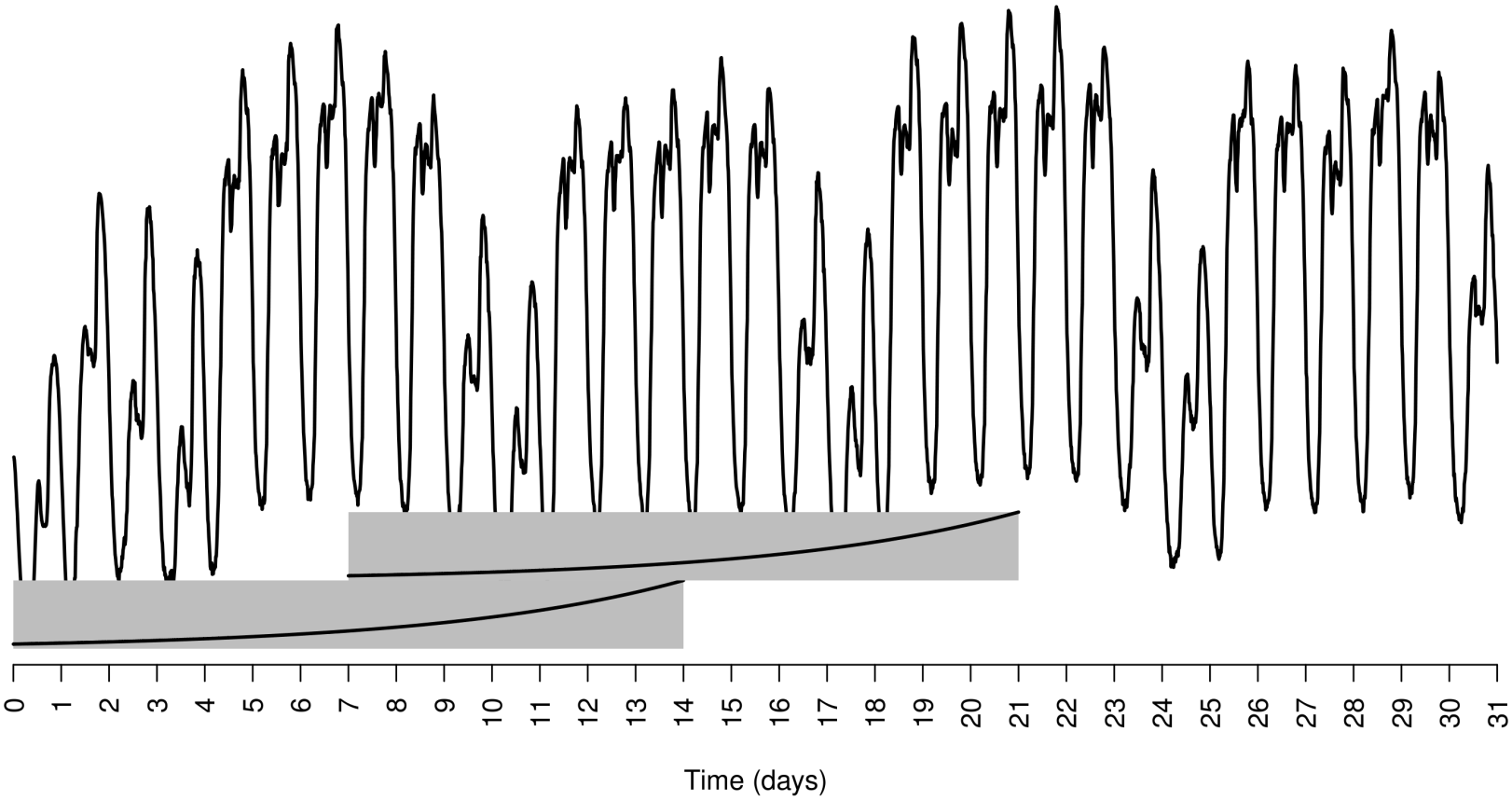
However, most simple methods use a catastrophic forget, that is, any past observation either is inside the window or it is not inside the window.

- Usually in streaming settings, the concept generating data evolves smoothly, so old data is less but still important.
- Even within the sliding window, the most recent data point is usually more important than the last one which is about to be discarded.

A simple approach considers giving weights to data points depending on their age within the sliding window.

- Several weighting models could apply: linear, loglinear, etc.
- Given its particular characteristics, we will present an exponential approach.

Alpha-weighed moving window



Alpha-weighed moving window

The main advantages of the α-weighted window are twofold:

- compared to traditional sliding windows, more importance is given to recent data points;
- compared to other weighting approaches, it can be maintained on the fly:

$$\dot{X}_{\alpha,w}(i) = \{x_i\} \cup \alpha \times \dot{X}_{\alpha,w}(i-1) \setminus \{\dot{x}_{i-w}\}$$

Alpha-weighed statistics

The α-weighted increment is

$$N_{\alpha,w}(i) = 1 + \alpha \times N_{\alpha}(i-1) = \sum_{j=1+i-w}^{i} \alpha^{i-j} = \sum_{j=0}^{w-1} \alpha^{j}$$

The α-weighted sum is

$$S_{x,\alpha,w}(i) = x_i + \alpha \times S_{x,\alpha,w}(i-1) = \sum_{j=1+i-w}^{i} \alpha^{i-j} x_j$$

The α-weighted average is

$$\bar{X}_{\alpha,w}(i) = \frac{S_{x,\alpha,w}(i)}{N_{\alpha,w}(i)}$$

Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream **over a alpha-weighed sliding window**.

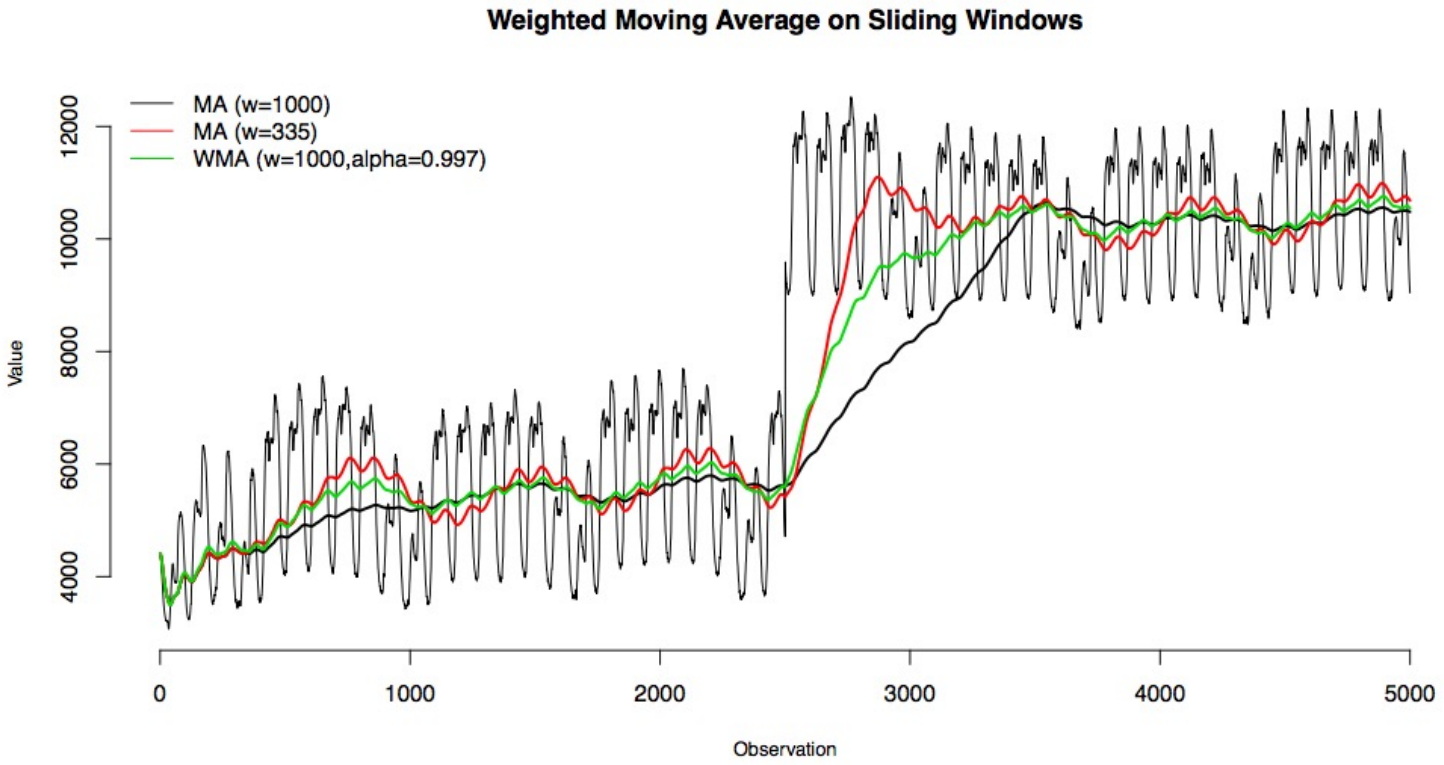
Exercise

Using the *stream* package in R, keep the current sample mean of a 1-dimensional data stream **over a alpha-weighed sliding window**.

```r
# keep sample mean over a alpha weighted sliding window

# definitions
itera <- 2000
set.seed(523)
stream1 <- DSD_Gaussians(k=1, d=1, mu=0, sigma=2)
stream2 <- DSD_Gaussians(k=1, d=1, mu=10, sigma=2)
w <- 100
eps <- 0.05
alpha <- eps^(1/w)

# initialization
n <- sum <- 0
plot(NULL, xlab="Iteration",ylab="Value", xlim=c(0,itera),
     ylim=c(-3,13), main="Weighted Sliding Mean")
window <- NULL

# iterative step
for (i in 1:itera){
  xi <- ifelse(i<=1000,
               get_points(stream1)[1,1],
               get_points(stream2)[1,1])
  sum <- sum * alpha + xi
  n <- n * alpha + 1

  window <- c(window, xi)

  if (length(window) >  w) {
    sum <- sum - window[1] * alpha ^ (w-1)
    n <- n - 1 * alpha ^ (w-1)
    window <- window[-1]
  }

  points(x=i, y=xi, pch=20, cex=.1, col="black")
  points(x=i, y=sum / n, pch=20, cex=.5, col="red")
}
```

Comparison

Approximates a small sliding window (better adaptation); keeps information from older data (smooth behaviour).

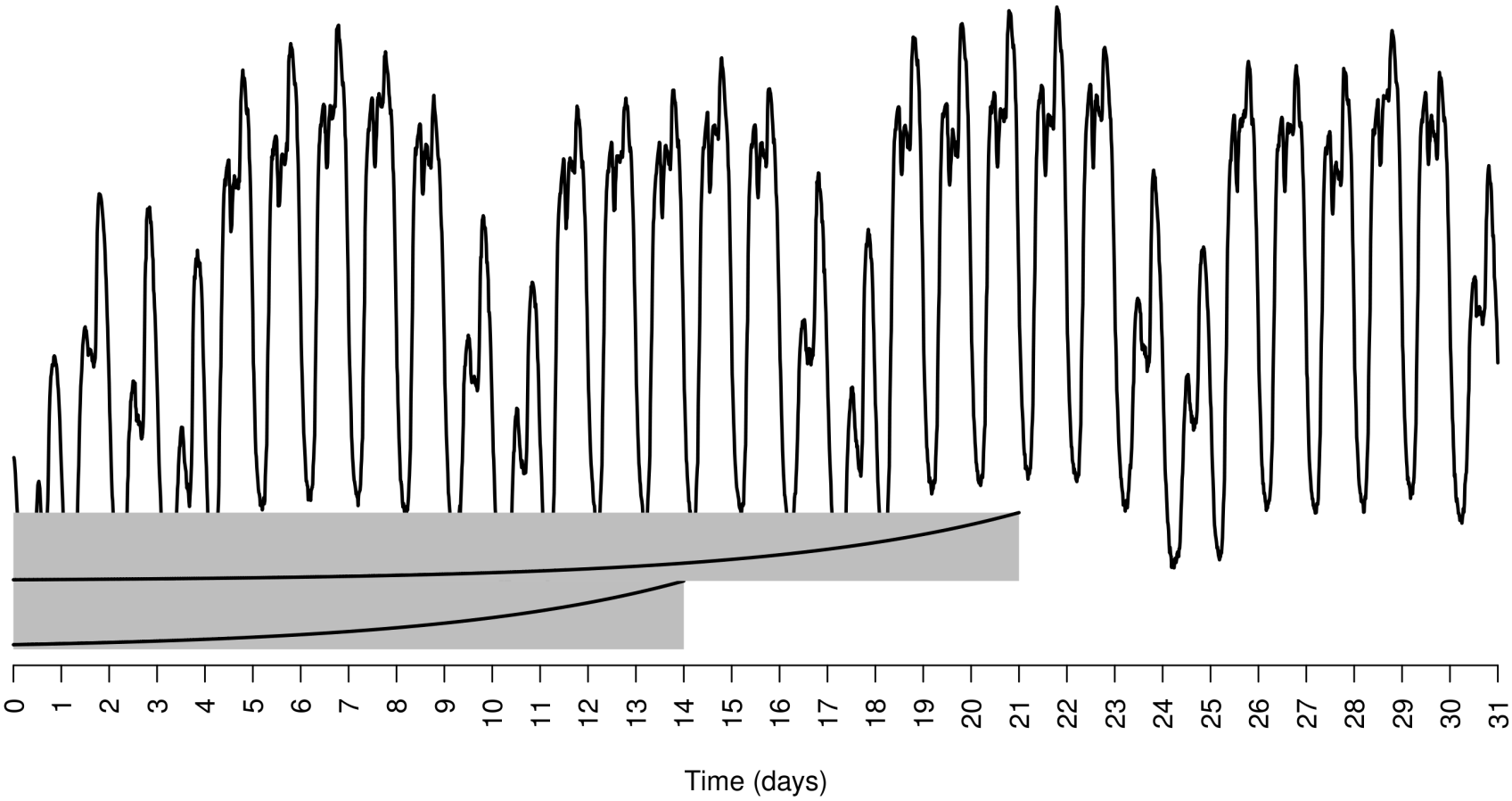


What if the window is too long?

- In ubiquitous streaming data sources resources such as memory and processing power are many times thin.
- Some times there is **not even enough memory to store all the data points inside the window.**

- We can make use of **fading windows**, a memoryless model which **uses all previous data to approximate a α-weighted window.**

Fading Window



Alpha-fading statistics

The α-fading increment is

$$N_\alpha(i) = 1 + \alpha \times N_\alpha(i-1), \lim_{i\to\infty} N_\alpha(i) = \frac{1}{1-\alpha}$$

The α-fading sum is

$$S_{x,\alpha}(i) = x_i + \alpha \times S_{x,\alpha}(i-1)$$

The α-fading average is

$$M_{x,\alpha}(i) = \frac{S_{x,\alpha}(i)}{N_\alpha(i)}$$

Exercise

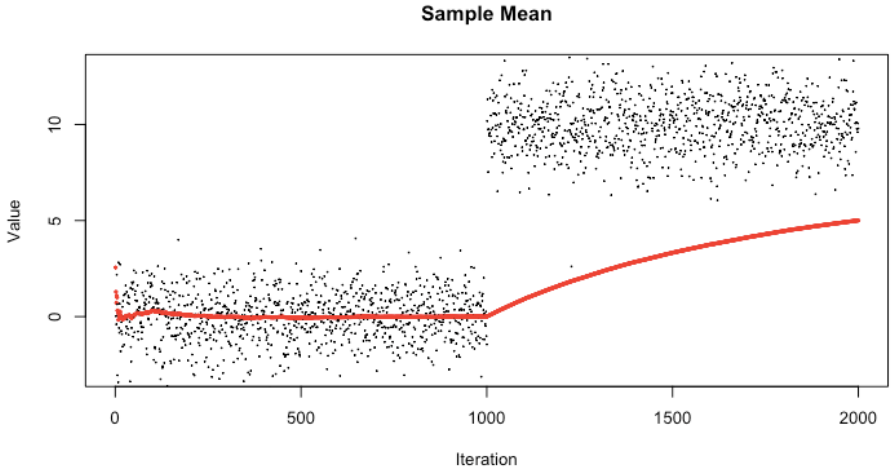
Using the *stream* package in R, keep current sample mean of a 1-dimensional data stream **over a alpha-fading window**.

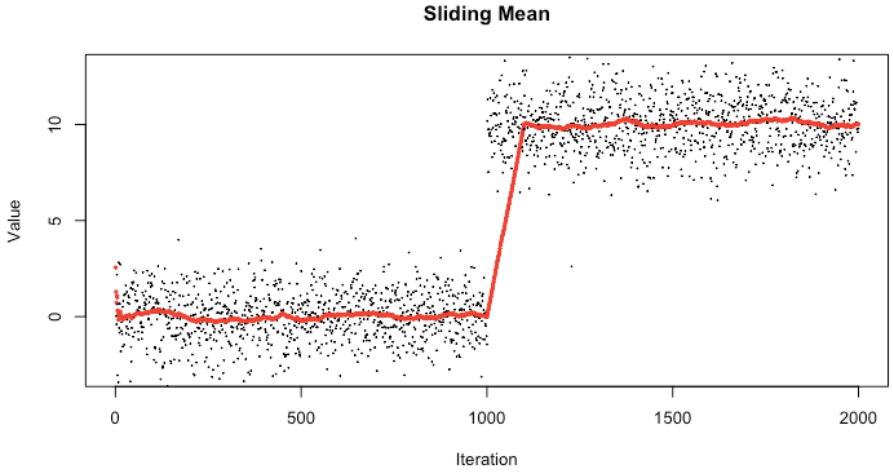
Exercise

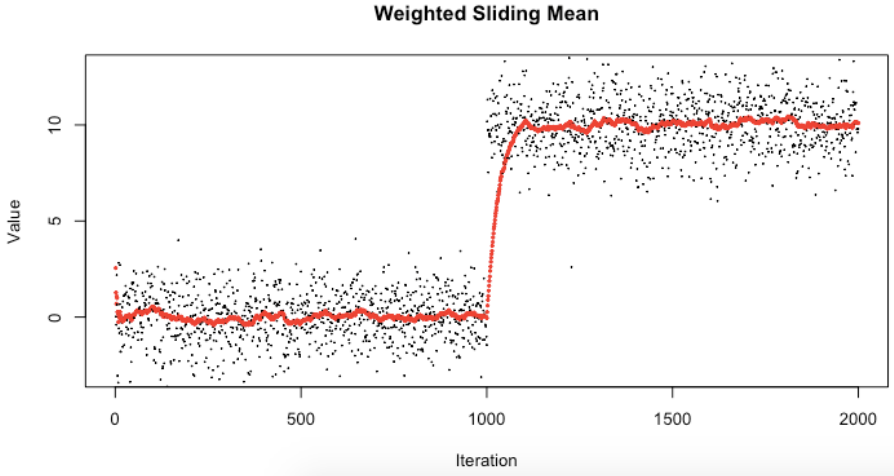
Using the *stream* package in R, keep current sample mean of a 1-dimensional data stream **over a alpha-fading window**.

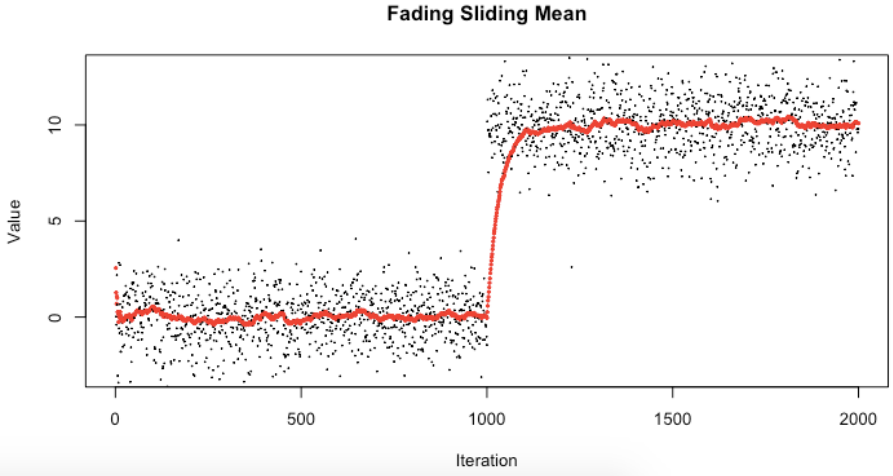
```
# keep sample mean over a alpha fading sliding window

# definitions
itera <- 2000
set.seed(523)
stream1 <- DSD_Gaussians(k=1, d=1, mu=0, sigma=2)
stream2 <- DSD_Gaussians(k=1, d=1, mu=10, sigma=2)
w <- 100
eps <- 0.05
alpha <- eps^(1/w)

# initialization
n <- sum <- 0
plot(NULL, xlab="Iteration",ylab="Value", xlim=c(0,itera),
     ylim=c(-3,13), main="Fading Sliding Mean")
window <- NULL

# iterative step
for (i in 1:itera){
  xi <- ifelse(i<=1000,
               get_points(stream1)[1,1],
               get_points(stream2)[1,1])
  sum <- sum * alpha + xi
  n <- n * alpha + 1

#  window <- c(window, xi)

#  if (length(window) >  w) {
#    sum <- sum - window[1] * alpha ^ (w-1)
#    n <- n - 1 * alpha ^ (w-1)
#    window <- window[-1]
#  }

  points(x=i, y=xi, pch=20, cex=.1, col="black")
  points(x=i, y=sum / n, pch=20, cex=.5, col="red")
}
```

Comparison

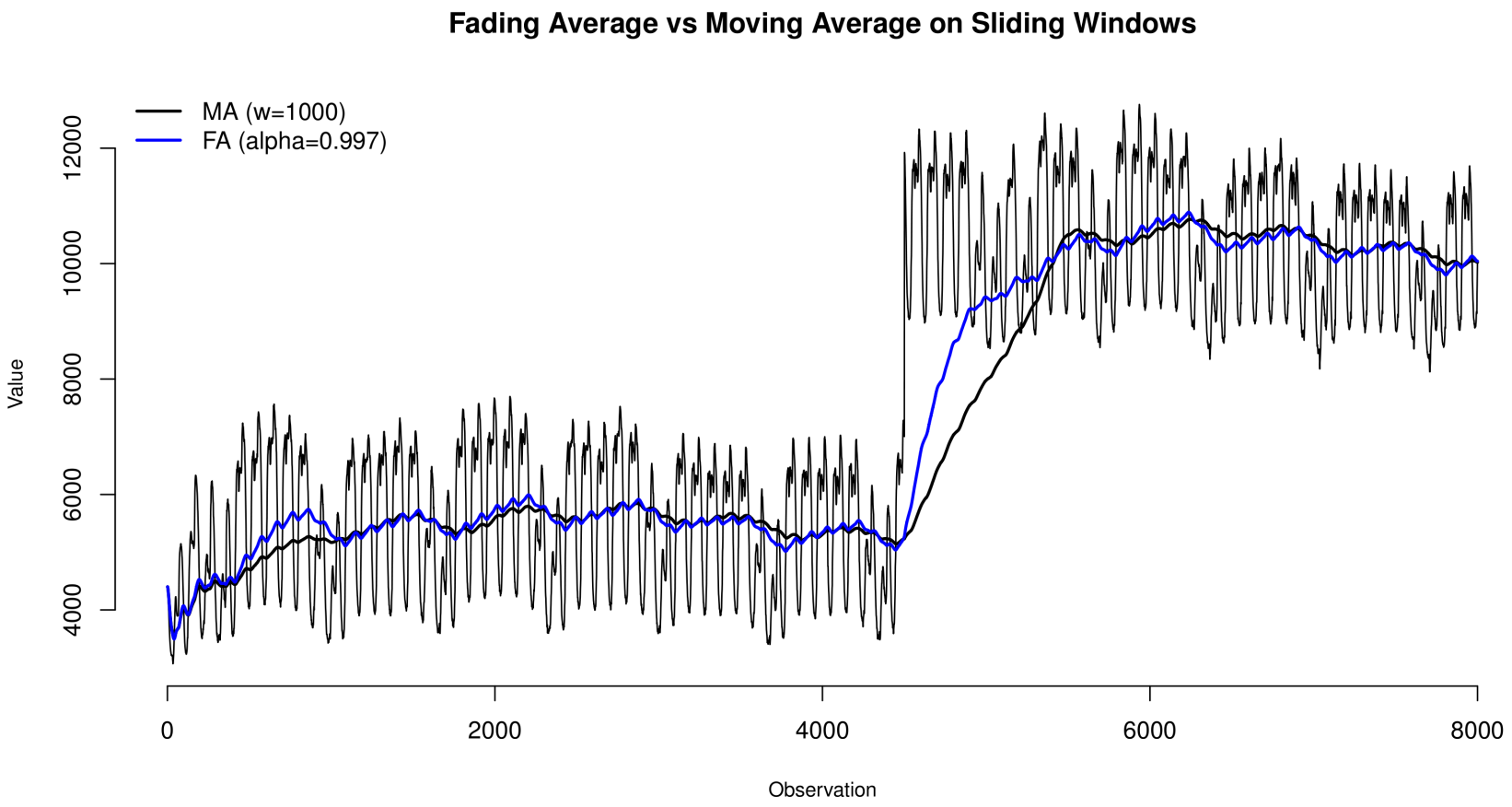

Sample Mean
Value
Iteration
0
5
10
0
500
1000
1500
2000


Sliding Mean
Value
Iteration
0
5
10
0
500
1000
1500
2000


Weighted Sliding Mean
Value
Iteration
0
5
10
0
500
1000
1500
2000


Fading Sliding Mean
Value
Iteration
0
5
10
0
500
1000
1500
2000

Fading Average



Approximating the Weighted Window Model

We are using **all previous data** in the computation of α-fading increments and α-fading sums.

To approximate a window model of size w, we need to check **how much of the data older than the objective w is being in fact used in this computation.**

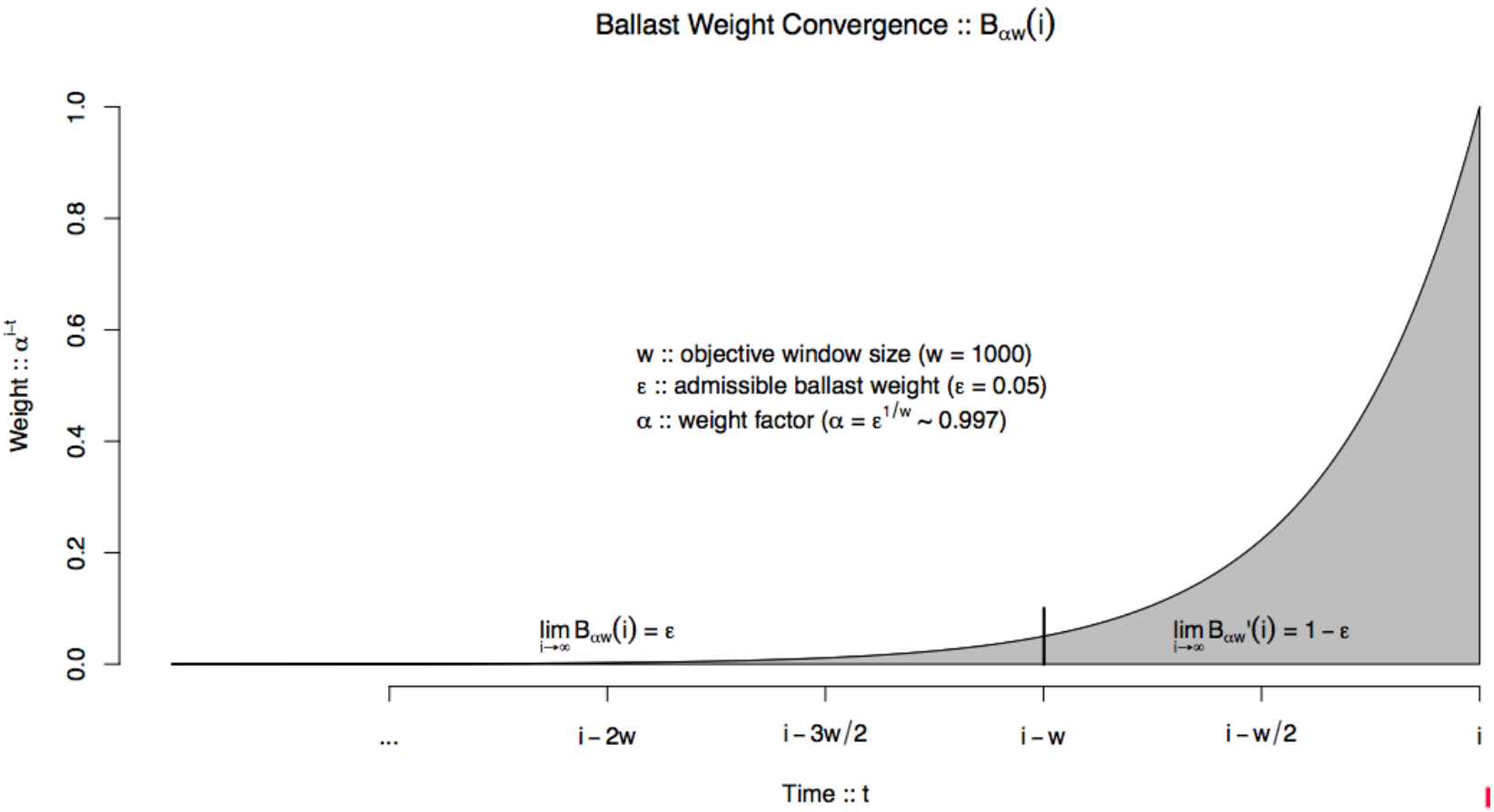
The **ballast weight** is the proportion of weight given to old observations (with respect to w) in the computation of the α-fading sum, i.e.

$$B_{\alpha,w}(i) = \frac{\sum_{j=w}^{i-1} \alpha^j}{N_\alpha(i)}, \lim_{i \to \infty} B_{\alpha,w}(i) = \alpha^w$$

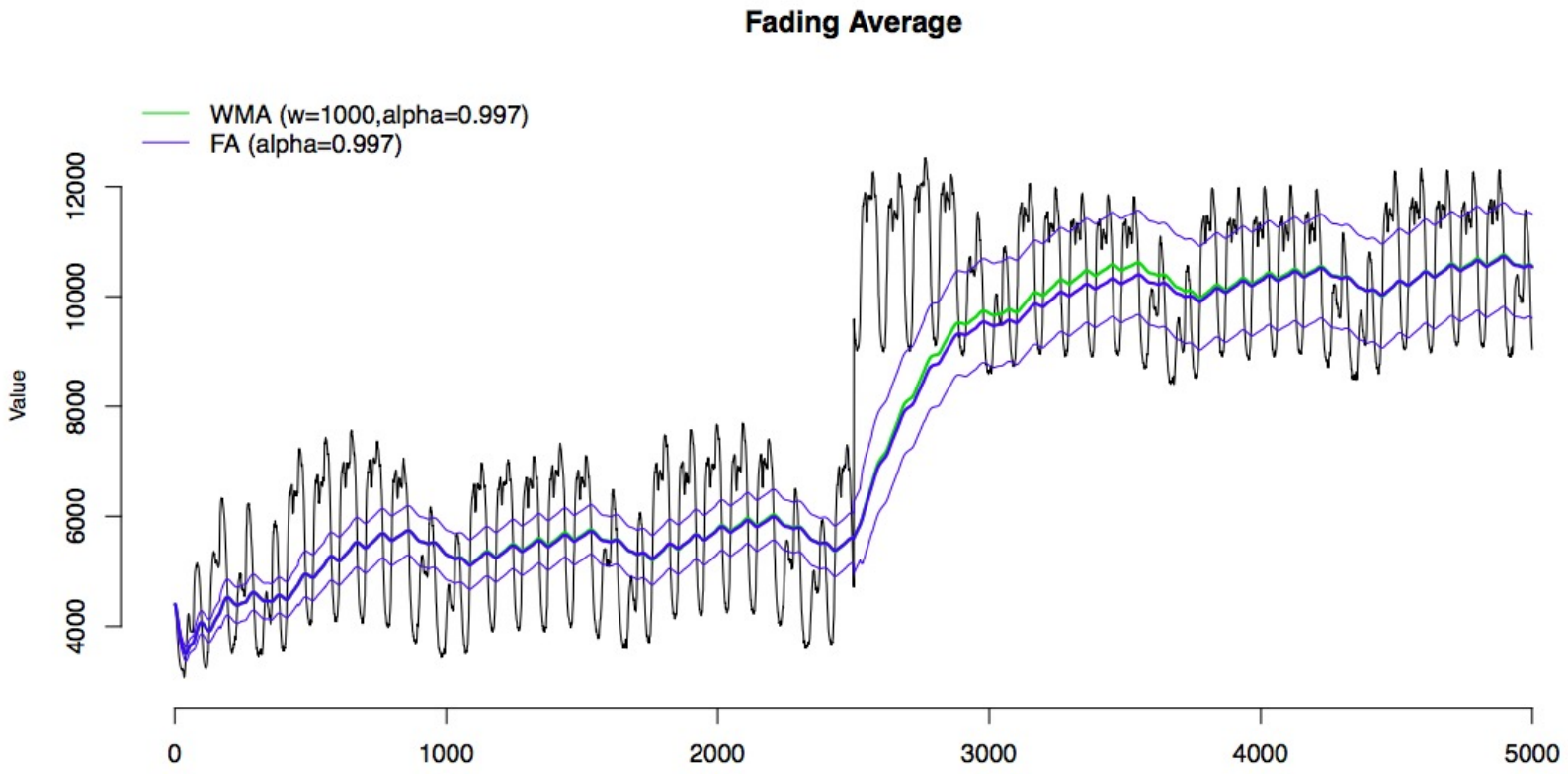
Given a fading factor α and an objective window size w, **we know exactly how much of the sum is based on data points older than w.**

We can define an admissible value for the ballast weight, so that we are approximating a sliding window with minimum error.

Approximating the Weighted Window Model



Approximating the Weighted Window Model



Alpha-fading statistics

We have a clear method to compute approximated weighted averages without keeping all the data in a sliding window.

- The α parameter can be properly defined in order to keep the approximation error within known bounds ($\alpha = \varepsilon^{1/w}$).
- Similar results can be produced for standard deviation, correlation coefficient and online histograms.
- A major advantage of this approximation is that it is independent of the distribution generating the examples, as no probabilistic assumption is made on the data.

Alpha-fading statistics

The α-fading variance is

$$V_{x,\alpha}(i) = \left\| \frac{S_{x^2,\alpha}(i)}{N_\alpha(i)} - \frac{S_{x,\alpha}(i)^2}{N_\alpha(i)^2} \right\|.$$

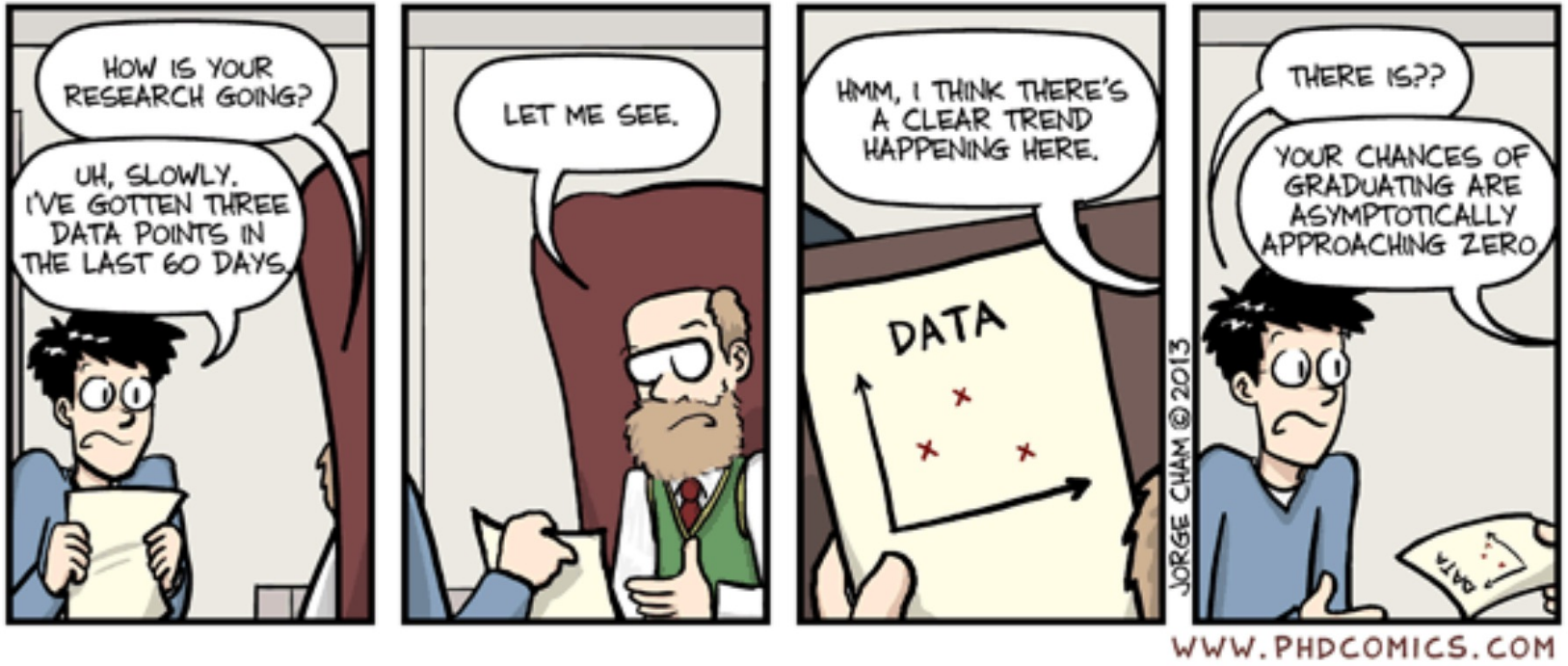
The α-fading correlation is

$$C_{x,y,\alpha}(i) = \frac{S_{xy,\alpha}(i) - \frac{S_{x,\alpha}(i) S_{y,\alpha}(i)}{N_\alpha(i)}}{\sqrt{\left\| S_{x^2,\alpha}(i) - \frac{S_{x,\alpha}(i)^2}{N_\alpha(i)} \right\|} \sqrt{\left\| S_{y^2,\alpha}(i) - \frac{S_{y,\alpha}(i)^2}{N_\alpha(i)} \right\|}}.$$

The α-fading histogram (interval frequency counts) is

$$F_{\alpha,l}(i) = c_{li} + \alpha \times F_{\alpha,l}(i-1),$$

A stream of data points…



Exercise

Using the *stream* package in R, keep current sample histogram of a 1-dimensional data stream using different window models.

Specifications:

- Data points from the stream must be accessed only once and used in the computation of sufficient statistics (e.g. sum or counters).
- Besides that, data points can only be additionally accessed if, using window models, we need to store the point in the window, case where another access is allowed in order to remove it later on from the window.
- At every new data point, a histogram needs to be available reflecting the distribution of recent data.